REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
OCT 08	Conference Paper Postprint	Oct 06 – Oct 07

4. TITLE AND SUBTITLE

DISTRIBUTED PLANNING IN A MIXED-INITIATIVE ENVIRONMENT COLLABORATIVE TECHNOLOGIES FOR NETWORK CENTRIC OPERATIONS

5a. CONTRACT NUMBER
In-House

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Chad C. DeStefano, Kurt K. Lachevet and Joseph A. Carozzoni

5d. PROJECT NUMBER
558S

5e. TASK NUMBER
IH

5f. WORK UNIT NUMBER
DP

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

AFRL/RISB
525 Brooks Rd.
Rome, NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RISB
525 Brooks Rd.
Rome, NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TP-2008-5

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release: Distribution is Unlimited. PA# 08-0046

13. SUPPLEMENTARY NOTES

Presented at the International Command and Control Research and Technology Symposium 2008,Bellevue,WA 17-19 June 2008

14. ABSTRACT

The USAF Command and Control (C2) is undergoing a transformation to enable a full-spectrum, joint warfighting capability. To be able to meet the future challenge of employing forces anywhere in the world in support of national security objectives, the USAF requires a highly synchronized, distributed planning and re-planning capability that is flexible to adapt to any level of conflict. This paper describes an in-house program underway at the USAF Research Laboratory Information Directorate that is developing technologies to support the concepts of Network-Centric Operations (NCO). The research focus of this program is on the concepts and architecture needed to support the distributed, mixed-initiative, planning required for NCO. Our system builds upon distributed blackboards and multi-agent systems to provide automated opportunistic planning capabilities for distributed C2 operations. An extensible UML model of plans has also been developed to support human-machine dialog for mix-initiative planning. The plan representation is object oriented recursive, and support plan fragment operations, a key concept for distributed planning.

15. SUBJECT TERMS
Distributed Command and Control, Analogical Reasoning, Network Centric Planning, Intelligent Agents

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Joseph A. Carozzoni
U	U	U	UU	23	19b. TELEPHONE NUMBER (*Include area code*) N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

13th ICCRTS: C2 for Complex Endeavors

"Distributed Planning in a Mixed-Initiative Environment"

Collaborative Technologies for Network Centric Operations

Authors

Chad DeStefano

Kurt Lachevet

Joseph Carozzoni

Point of Contact:

Chad DeStefano

Air Force Research Laboratory Information Directorate

Systems & Information Interoperability Branch (RISE)

525 Brooks Road, Rome, NY 13441-4505

315-330-4286

Chad.DeStefano@rl.af.mil

Postprint

Distributed Planning in a Mixed-Initiative Environment

Abstract

The USAF Command and Control (C2) is undergoing a transformation to enable a full-spectrum, joint warfighting capability. To be able to meet the future challenge of employing forces anywhere in the world in support of national security objectives, the USAF requires a highly synchronized, distributed planning and replanning capability that is flexible to adapt to any level of conflict. This paper describes an in-house program underway at the USAF Research Laboratory Information Directorate that is developing technologies to support the concepts of Network-Centric Operations (NCO). The research focus of this program is on the concepts and architecture needed to support the distributed, mixed-initiative planning required for NCO. Our system builds upon distributed blackboards and multi-agent systems to provide automated opportunistic planning capabilities for distributed C2 operations. An extensible UML model of plans has also been developed to support human-machine dialog for mix-initiative planning. The plan representation is object oriented, recursive, and supports plan fragment operations, a key concept for distributed planning. This paper will also discuss our future research directions, including the encoding of human-planner experience and expertise for rapid formation of distributed expert planning teams.

Keywords: Distributed, Planning, Mixed-Initiative, Network-Centric Operations (NCO), Distributed Blackboard, Multi-Agent System

1 Introduction

1.1 Problem Statement

The U.S. and other highly industrialized nations have developed military capabilities that excel in conventional force-on-force warfare, especially where tactics are well developed and known. However, modern adversaries have devised the strategy of not going "head-to-head" with these capabilities and instead combat modern conventional forces with unconventional tactics. One example of the result of a weapon system being vastly superior is the case of the air superiority fighter which modern adversaries totally avoid putting themselves in a position to contest them.

To meet these future challenges, U.S. forces are in the midst of a "transformation" to not only support traditional high-tempo, large force-on-force engagements, but also smaller-scale conflicts characterized by insurgency tactics and time-sensitive targets of opportunity. This transformation requires a vastly new Command and Control (C2) process that can adapt to the any level of conflict, provides a full-spectrum joint warfighting capability, and can rapidly handle any level of complexity and uncertainty.

1.2 Future C2 Requirements

To meet future challenges, the U.S. Air Force (USAF) is moving towards a model of continuous air operations not bounded by the traditional 24-hour Air Tasking Order (ATO) cycle. Meeting these objectives will require a highly synchronized, distributed planning and replanning capability. As a potential way ahead, AF/A5 (Plans) in May 2006 released a revolutionary vision paper titled "C2 Enabling Concepts" depicting what a potential future C2 environment could be. Four key concepts emerged from this vision of a future AOC:

- Distributed/Reachback planning
- Redundant/Backup planning
- Continuous planning
- Flexible, scalable, tailorable C2

Experience with recent operations also reveals that the C2 process must transition from a process of observation and reaction to one of prediction and preemption. To achieve this, we will need to go beyond the focus of military operations, and instead address the entire spectrum of Political, Military, Economics, Social, Infrastructure, and Information (PMESII).

1.3 Information Age C2 Solutions

The focus of this research has been founded on two emerging concepts for the future of C2. Developing a C2 environment that supports the vision of Network Centric Operations (NCO) was task number one. The tenets of NCO are:

- Information sharing
- Shared situational awareness
- Knowledge of commander's intent

1.4 Objective of the DEEP Project

The long-term goal of the Distributed Episodic Exploratory Planning (DEEP) project is to develop in-house a prototype system for distributed, mixed-initiative planning that improves decision-making by applying analogical reasoning over an experience base. The two key objectives of DEEP are:

- Provide a mixed-initiative planning environment where human expertise is captured and developed, then adapted and provided by a machine to augment human intuition and creativity.
- Support distributed planners in multiple cooperating command centers to conduct distributed and collaborative.

The architecture of DEEP was explicitly designed to support these tenets of NCO in a true distributed manner. Because DEEP is not based on any current C2 system, we are able to explore concepts such as combining planning and execution to support dynamic replanning, machine-mediated self synchronization of distributed planners, and experiment with the impact of trust in an NCO environment (i.e. "Good ideas are more important than their source").

1.5 Research Areas

Alberts and Hayes (2007) advocate bold new approaches beyond current organizational process, focusing on what is possible for NCO. High priority basic research topics recommended as areas to systematically explore are:

1. Taxonomy for planning and plans;
2. Quality metrics for planning and plans;
3. Factors that influence planning quality;
4. Factors that influence plan quality;
5. Impact of planning and plan quality on operations;
6. Methods and tools for planning; and
7. Plan visualization

Pursuant to achieving the vision of DEEP, essentially all the above topics needed to be addressed. The first topic was the starting point and has received the most attention. The earliest effort in support of distributed planning was on CPR, an object-oriented plan framework developed under the ARPA-Rome Laboratory Planning Initiative (ARPI). CPR is based on the Unified Modeling Language (UML) which is well suited as the human-machine dialog to support mixed-initiative planning. The recursive nature of CPR supports multi-level planning at all levels (strategic, operational, and tactical), along with

plan fragments supporting distributed planning on a plan simultaneously. A key research topic for DEEP in 2008 is maintaining referential integrity when distributed planners simultaneously work on multiple sub-plans and/or plan fragments of a larger plan.

2 DEEP

2.1 High Level Architecture Overview

DEEP is a systems-of-systems architecture (Figure 1), comprised of the following systems:

- Distributed Blackboard for multi-agent, non-deterministic, opportunistic reasoning
- Case Based Reasoning system to capture experiences (successes and/or failures)
- Episodic Memory for powerful analogical reasoning
- Multi-Agent System for mixed initiative planning
- ARPI Core Plan Representation for human-to-machine common dialog
- Constructive Simulation for exploration of plausible future states

In describing the components of the architecture, DEEP contains a messaging system, data objects, shared data storage, and a number of agents, both interface and critic. To describe the pieces in the architecture, the order in which the system is used will provide the structure for the description. Along with explaining the DEEP architecture, this paper will discuss our proof-of-concept prototype architecture implementation. This prototype serves as our research platform to extend much more fully in the coming years.

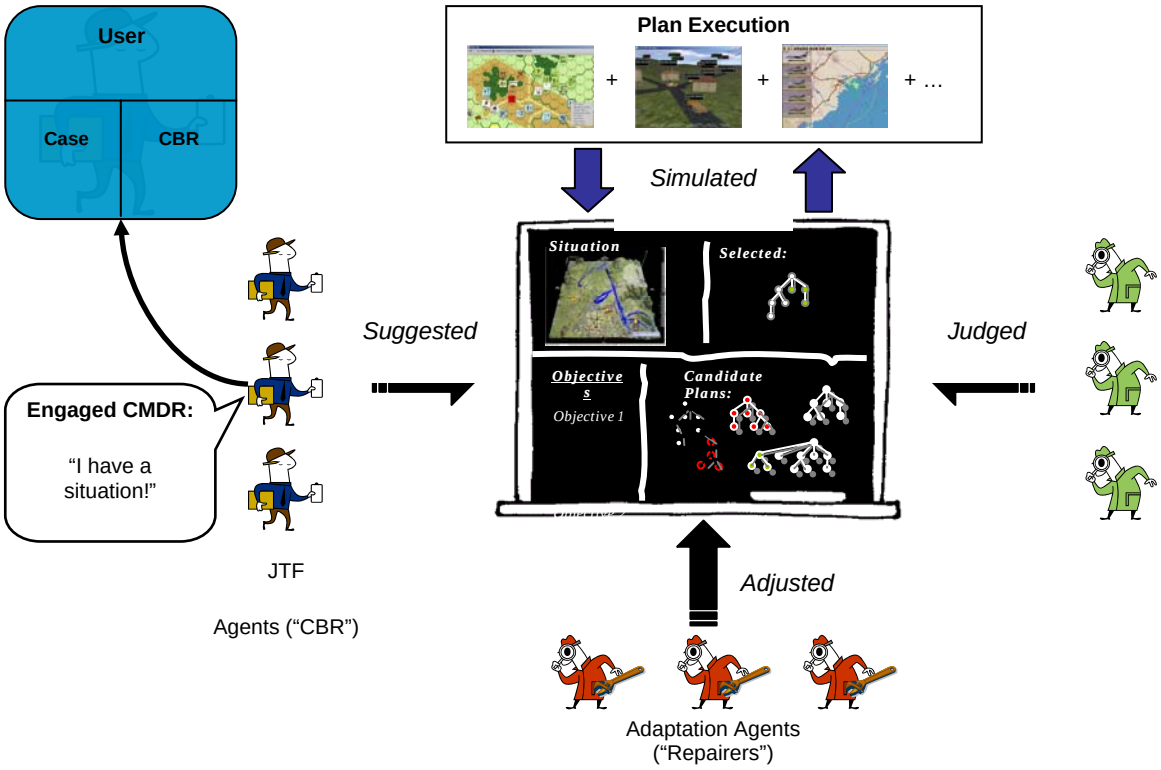

Figure 1 - DEEP Architecture

The process will now be described referencing Figure 1. The starting point for entry into the system is the commander using a planning agent (1). The planning agent allows for the commander to input information into the system which defines their current objectives. These objectives, along with other information, such as resources, locations, and time constraints, are collectively known as the situation. This situation is then placed on the blackboard (2). The blackboard then notifies all registered components of the new situation. The other planning agents, with their associated case bases and cased-based reasoning, search their case base using the situation given for relative past experiences (3). These results are then modified to fit the current situation (4) and are posted to the blackboard (5). Once the "candidate plans" are on the blackboard, they are adapted by specialized agents to further refine these plans (6). These plans are now ready for critique by the critic agents. These agents concurrently scrutinize the plans and score them based on their individual expertise (7). Once the plans are scored, the execution selection critic gathers the adapted plans along with their scores, determines their overall scores, and selects a number of top rated plans to be executed (8). The top rated plans are now run against a simulation (9). Now that there is a new plan which has been simulated, the results of this simulation are assimilated with the plan and stored in the casebase. The process is not over at this time, but instead the architecture can allow the plans to be run through the cycle many more times if desired (10). This was a simple explanation of the way the architecture works and an introduction to most of the components of the system. In addition to the specific critic agents, there are also other mechanisms and components

to the system which allow all of the component interaction to occur, such as messaging and specialized data objects. Before we discuss the separate components of the architecture, we will explain the methods and objects which facilitate the process described previously and will now be explained.

2.2 Component Interaction

Being a network centric application, there is a strong emphasis in DEEP on how data is passed and how that data is encapsulated. Before discussing the major structural components of DEEP, the paper will present a discussion on how the components interact and on the data object the components process. The next two sections will discuss the data object used in DEEP and the DEEP messaging system.

2.2.1 Plan Representation Storage

To facilitate the working of the components in DEEP, there needs to be an object which stores the data. This object also has to be well defined so it can be understood by all the DEEP software components. DEEP presumes to research the future of planning not just for the Air Force, but to be indifferent to the planning outfit or planning level. DEEP needs to be able to handle plans that are joint operations and handle plans at different levels. By different levels this means planning strategically, tactically, or operationally. The aspect of being joint also means that the plan representation has be able to display the semantics of terms. This is a topic of future research for DEEP and is discussed later in the report in the future challenges section. Due to the machine reasoning need of the DEEP system, the chosen plan representation also has to be easily machine readable.

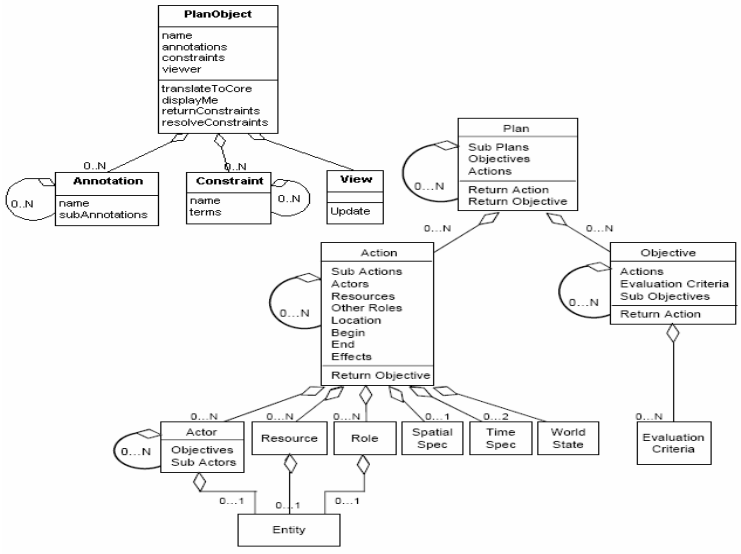

Figure 2 - Core Plan Representation

Some comparisons of different options were made and the selected plan representation chosen was the Core Plan Representation (CPR), seen in Figure 2. CPR is an object oriented structure agnostic to the planning level, which could be strategic, tactical, or operational (Pease 1998). Its natural object oriented structure also lines up very well with the machine reasoning capability DEEP needs to provide. The DEEP team adapted the original CPR structure (Figure 2) to meet the needs of DEEP and has evolved over time as required.

In DEEP, CPR represents case experience, which is composed of a plan, events, and an outcome. The attributes that are part of the plan are what the similarity metrics that are part of the cased-based reasoning component process over. Simulation of the plan populates the events and outcome sections. DEEP CPR has changed greatly from the structure shown in Figure 2 and will continue to evolve with DEEP as the program progresses.

CPR in its instantiation is used extensively throughout the DEEP architecture, but to do this a formalized messaging system is also required. This messaging system will be discussed next.

2.2.2 System Messaging

As just discussed, CPR is involved in vast amounts of component activity throughout the DEEP architecture. This network traffic requires a formalized messaging model for the component interaction required by the system.

The components that interact are all types of agents in the system and the blackboard. The architecture supports and requires inter-agent communication. Messaging is required to support actions such as an agent notifying the interface agent that the system requires input from the user. To accomplish this, a formalized messaging scheme is required with a defined structure so that new components are able to understand incoming messages as well as transmit their own. The DEEP Architecture includes a formal messaging scheme to be used by the other components.

In the current DEEP architecture, the communication protocol is supported by the publish-subscribe communication paradigm. At a high level, components (including agents) subscribe to the blackboard and are notified of new publications. Because of the push to create a functional proof-of-concept architecture, a simple taxonomy is currently in place to determine notification and message types until a more formalized communications protocol is established. The blackboard mediates all messaging in that it uses the blackboards defined messaging scheme and connectivity medium. The general scheme is that an agent in the system registers itself to the blackboard and then becomes a listener to blackboard messages. The blackboard at this time has a simple protocol of different types of messages that it operates on. There are default blackboard messages that it uses for communication, especially like placing an object on the blackboard or taking one off. Agents themselves do not communicate, but instead use the blackboard as a hub of communication. Due to this design philosophy early on, the blackboard also supports message forwarding to be utilized by agents to send special messages. An

example of this is that the interface planning agent needs a special message to know it has to ask the user to select out of the best chosen plans. This is also an example of the mixed-initiative interaction involved in the DEEP cycle.

Here is a further example of how messaging occurs in the current DEEP architecture implementation. Let us assume a simple setup of two planning agents, one blackboard, one critic, and one simulator. All components are registered to the blackboard for communication. 'Planning Agent A' becomes an engaged agent and a commander interfaces to it and inputs a situation, which is then placed on the blackboard. Once on the blackboard, the blackboard system notifies registered components with a message indicating the type of object by broadcasting these messages. The notified components have to decide if the packet sent down is of use to them. This happens for all communication, so in our situation when 'Planning Agent A' sends up a situation, all registered components get the packet including critics who cannot do anything with a situation. This is how the system continues to functions throughout and is how the simulator knows not to process until critic agents have had their processing time.

The previous pieces can be thought of as the "glue" used in the DEEP system to interface the various components. The previous examples have highlighted the components to be discussed next.

2.3 Architecture

Application interfacing having been discussed, it's now time to explain the components of the DEEP architecture, which are the distributed blackboard, interface agents, and critic agents. The explanation will provide an understanding of how the pieces fit into the mixed-initiative distributed architecture of DEEP.

2.3.1 Distributed Shared Data Structure

As you can see from the DEEP architecture diagram in Figure 1, the various DEEP components rely on a shared data structure to act as a medium of communication and interaction. Also, in order to support the new distributed Air Force explained earlier, the DEEP architecture requires a mechanism that supports reach-back in a distributed system. A blackboard system was chosen to fulfill this need as it not only functions as a shared memory for the DEEP components, but provides other functionality as well.

A blackboard system is an opportunistic artificial intelligence application based on the blackboard architectural software engineering paradigm (Corkill). In DEEP, the blackboard system is needed to function as a central data store to facilitate the communication and interaction between the different software components, including the interface agents, critic agents, and simulation engines which will be explained further later on. This interaction is made possible by the sharing and passing of objects.

Current commercial and open source blackboard system implementations are not distributed, so the paradigm needed to be extended from a monolithic to a distributed environment. To fully meet the requirements of DEEP, a distributed blackboard system was designed and implemented.

Traditionally, a blackboard consists of three discrete components: the blackboard data structure which is a central repository for data objects, the knowledge sources which are specialist software modules (agents in the DEEP software architecture) which provide specific expertise required by the system, and lastly is a control component which controls the flow of objects and problem-solving activity in the system. (Corkill)

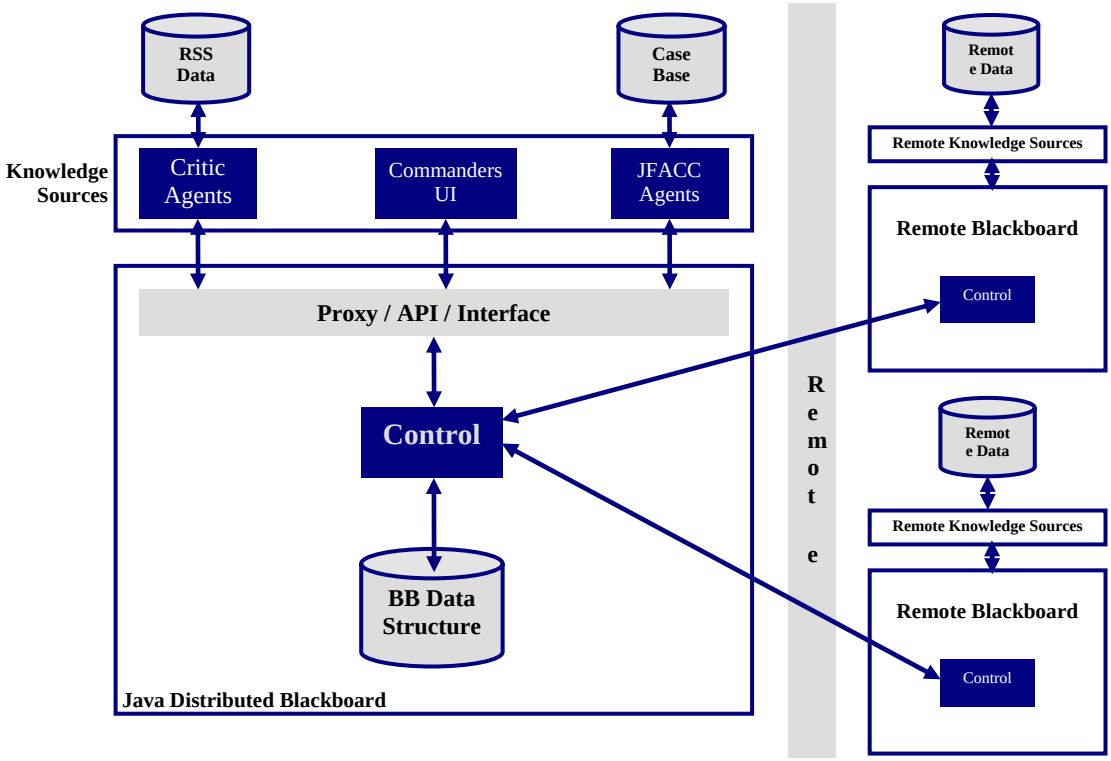

Figure 3 - Distributed Blackboard Architecture

2.3.1.1 Core Data Structure

The core data structure (see Figure 3) of the blackboard is the global data store which holds the objects within the system and is accessible by all of the system's knowledge sources. Because there could potentially be an extremely large number of objects placed on and contained within the blackboard at one time, blackboard data structures are conventionally divided in more than one way. These divisions are known as panes and layers, and could potentially contain further dimensions of separation. (Corkill)

In the DEEP architecture, the core data structure is defined to provide certain functionality. The data store component of the blackboard has been abstracted out to allow for future revisions and extensions to how and where the data is stored. This interface allows the option for the backend of the blackboard to become a database or other high performance data store.

In DEEP's current state, the core data structure is a matrix of Hashtables. These Hashtables within Hashtables allow panes and layers to be identified by providing a unique key and utilized to divide the blackboard up into more manageable sections. Now

that there is a global repository for data, software components may start to utilize it to solve problems.

2.3.1.2 Knowledge Sources

By simply connecting to the blackboard, an application has the ability to become a knowledge source of the blackboard. Figure 3 shows example knowledge sources from the DEEP architecture and an example of how a knowledge source could contribute external information to the blackboard. A key characteristic to knowledge sources in a blackboard system is that they require no knowledge of the other knowledge sources that are connected to the blackboard. They bring their specialized expertise to the system and do not rely on others to provide it. Each knowledge source is responsible for knowing when, what, and how it may contribute to the solution to the current problem on the blackboard. (Corkill)

In the DEEP architecture, a component must implement an interface provided by the blackboard in order to connect to it. The connection process includes connecting to the local blackboard proxy and registering with the blackboard for blackboard update event notifications (more on this in the control discussion). So now there is a data store and knowledge sources are ready to place objects on it. The next section discusses the control mechanism of the blackboard to facilitate the problem solving which will take place on the blackboard.

2.3.1.3 Control

There are several control system paradigms which may be employed when designing a blackboard system. It may be very centralized to the blackboard, distributed amongst the blackboard and knowledge sources, or pushed out to the knowledge sources, requiring them to facilitate their own control of contributions to the problem. (Corkill) The blackboard system developed for the DEEP architecture splits the control between the blackboard application and its knowledge sources. The control component on the actual blackboard application side directs communication amongst the distributed blackboards, whereas the knowledge sources are held responsible for choosing whether or not they should interact with new or updated objects on the blackboard, or even taking initiative and placing a new object on the blackboard without waiting for a blackboard event notification.

The control component of the blackboard is also what supports multiple blackboards to remain synchronized and distributed. Because the control component manages all of the activity occurring within the blackboard system, it is able to control how information is distributed amongst the connected blackboards applications and remain synchronized through the use of queues and messaging schemes. This is what allows there logically to be a single distributed blackboard while physically there are multiple, synchronized replicated blackboards. When a new object is passed to the blackboard proxy by a knowledge source, it is passed through the control mechanism which distributes it to all connected blackboard applications. After all the connected blackboard systems receive

the new object, they are placed in their local data store waiting to be manipulated or retrieved.

In addition to the three traditional blackboard components, the Distributed Blackboard designed for DEEP includes additional components which are necessary for a distributed blackboard and uniform communication between the knowledge sources connected to the distributed blackboard system.

2.3.1.4 Proxy

The proxy is an interface provided by the blackboard in which a knowledge source connects to and interfaces with. This proxy connection is established using a network socket. Originally, a blackboard application was designed to be running on each computer that contains one or more knowledge sources; however, because the knowledge source connects using this network socket, it may reside on a separate computer.

This proxy allows the interface to perform actions to the blackboard such as the putting and retrieval of objects. Other actions include the retrieval of an object by its unique identifier and the registration of new blackboard listeners. Similarly to the core data structure, the proxy interface could easily be extended to accommodate integration with other applications (new or existing) as needed.

2.3.1.5 Blackboard Objects

A well defined common interaction language is also necessary for a successful blackboard system. To keep the distributed blackboard as flexible as possible, the blackboard provides a simple interface to the knowledge sources for objects to be placed on the blackboard. This interface forces objects placed on the blackboard to contain certain properties and functions so the blackboard can work with the object. The properties include the partition the object belongs to, the UID of the object, as well as a timestamp. By implementing this interface, the object also becomes serializable, meaning it may be transmitted over a network socket.

2.3.1.6 Blackboard Utilities

A main utility in the blackboard utilities is the Packet. This packet is utilized by the control to send messages to other connected blackboards and is what the knowledge sources receive when they get an update event from the blackboard. Depending on the packet type, it contains certain useful information, some containing blackboard objects.

Another blackboard utility is the BBUID, which is a unique identifier across a network. This UID is required for all blackboard objects, components, and knowledge sources. There are also other convenience utilities such as a log writer and properties file parser.

The Distributed blackboard is an integral part of the DEEP architecture in that it provides the functionality and ability for the system to become distributed. Now that there is a distributed data structure, we can look at the DEEP knowledge sources and how they will utilize the blackboard.

2.3.2 DEEP Agents

After having discussed the blackboard, the shared data structure, the agents connecting to it shall be explained. The DEEP system uses two different types of agents. The first type is called an interface agent or planning agent and the second type is a critic agent.

Agents in the DEEP architecture extend and use the Java Agent Development (JADE) framework. DEEP requires a distributed multi-agent system and a framework to help simplify the implementation of this system. JADE was chosen because it is fully implemented in Java, and supports these requirements.

2.3.3 Interface Agents

Interface agents are the interfaces through which a planner will input a situation and communicate with the DEEP system. Planning agents are the wrapper for the casebase, utilize case-based reasoning on the casebase, and are the user front end for the planner and allow for mixed-initiative interaction with the system.

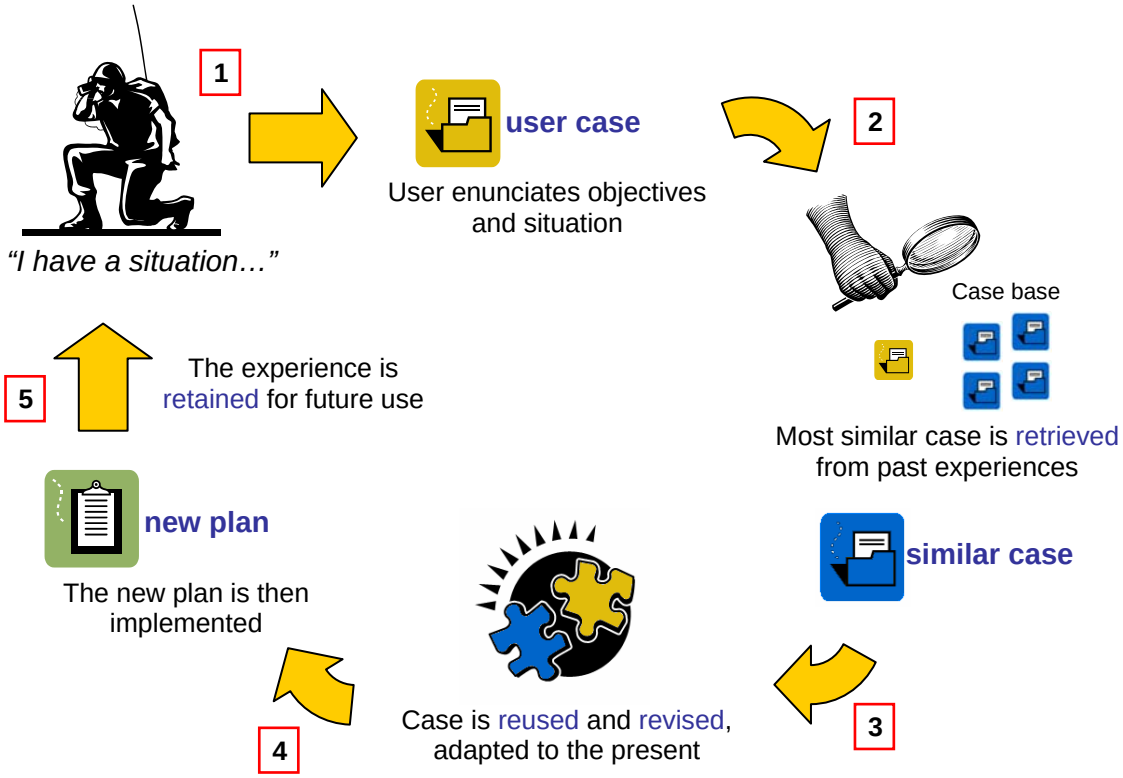

Figure 4 - Case Based Planning

DEEP uses jCOLIBRI (http://gaia.fdi.ucm.es/projects/jcolibri/), an object-oriented framework in Java for building Case Based Reasoning (CBR) systems. Figure 4 illustrates how the Case Based Reasoning cycle is applied to Case Based Planning and used in the DEEP architecture. Case-based planning makes use of past experiences to

implement new plans and retain their outcomes. “Case-based planning is the idea of planning as remembering.” (Hammond)

The planning agent allows the user of the system to input into the planning agent a situation using a user interface. Once the operator feels comfortable with the input, the agent, via the user interface, allows the situation to be forwarded to the blackboard. The situation includes objective statements, locations, actors, resources, and times. This is the primary way the system allows user interaction. While operating with the user interface, the operator can also view plans on the blackboard if there are any and view the casebase associated with the planning agent. The casebase for each planning agent will be unique.

Once a situation has been placed on the blackboard, the blackboard will broadcast a message out letting all components registered know about a data change with an identifier. The listener in the planning agent finds out what type of object was placed on the blackboard. Planning agents in the system react to a new situation because a situation initiates cased-based reasoning for the planning agent.

The case-based reasoning component will now occur since a situation has been presented to the system. This will not be discussed in depth because it was demonstrated by Anthony Ford at ICCRTS 2007 in (paper 078). Using the situation, a user defined number of cases are retrieved and placed onto the blackboard. Once the plan(s) are placed on the blackboard, the plans are processed by the critic agents and will be discussed in detail in the critic agent section.

The casebase in which the plans come from is generally unique to each planning agent and in our case each planning agent represents the experience of some entity or group of entities, namely commander experiences. The casebase of an entity can contain experiences of any kind, and this fits in with the CPR data object and its ability to work with different planning types.

In terms of technology, the planning agent provides a user interface to the user, interfaces to a casebase, and interfaces with a reasoning engine. These interfaces are important due to interchangeability of these components. Little processing is done by the planning agent itself, and instead by external components that it interfaces with. The agent itself is the medium between the reasoning process and the blackboard and the human and the blackboard. Now that the plans are on the blackboard and ready for evaluation to make sure of their correctness, it is time to discuss the critic agents.

2.3.4 Critic Agents

The term critic agent is used to describe a number of agents in DEEP that evaluate plans based on a number of criteria. The criteria can be loosely coupled into the following categories; adaptation, critic, execution, and simulation. Adaptation critics are plan refiners. Critic agents are evaluators of plans that score based on their own inherent knowledge. Execution critics determine ongoing plans in the DEEP cycle. Simulation critics simulate plans for various reasons. All of these different types of critics are going to be discussed in more depth in the following paragraphs.

2.3.4.1 Adaptation Critic Agents

Adaptation Agents in the DEEP architecture are software components which specialize in further refining a plan based on their particular area of expertise. As explained earlier, the initial plan that is instantiated to the new situation and placed onto the blackboard is a "rough cut" and needs supplementary revision. When the adaptation critic agent receives notification from the blackboard that there is a new instantiated plan on the blackboard, it reviews the plan, makes its changes, and posts a new version of the plan with its adaptations.

There are many possibilities for different adaptation agent specializations. For a proof-of-concept, a Capabilities Adaptation Agent was designed, developed, and integrated into the DEEP architecture. The Capabilities Adaptation Agent's specialization is validating that the actors in the instantiated plan are capable of performing the actions they were assigned to. In order to accomplish this, it first has to be determined what roles an actor is capable of performing and validate that it is consistent with the action it has been assigned to. Actors in the DEEP architecture have both default roles as well as specialized roles for the situation that need to be taken into account. If this actor is not capable, a new actor must be found to replace it. The plan then looks in the situation at available actors and uses the similarity metric explained earlier to look for a similar actor. This new, similar actor then replaces the incapable actor in the action. After the agent has adapted this plan using its specialized knowledge, it then posts it to the blackboard.

Adaptation Agents are also an area of possibility in which mixed-initiative interaction can once again be brought into the system. Interfaces could be made which would allow the plan to be displayed to a user, either as a whole or in a specific way, which would allow the user to apply their expertise and adapt part of the plan.

So far a scenario has been created by specifying objectives, methods and resources which was then placed on the blackboard. Each planning agent has then looked into their case base to find similar situations which were similar and used them to instantiate a new plan based on their past experience. Now that the plans on the blackboard have been instantiated, and refined by multiple Adaptation Agents, they are ready to be scored and criticized by the Scoring Critic Agents.

2.3.4.2 Scoring Critic Agents

This category of agents can be quite extensive, but the present DEEP system only uses one for demonstration purposes. The particular one used was a weather agent, but the future possibilities include political, logistics, ethical, legal and cyber agents among others.

Scoring agents focus themselves on certain areas of a plan. Weather agent would focus on how weather impacts the plan and not stress political fallout. A legal agent would not focus on weather, but instead focus itself on the legal aspects of the plan. Due to this, the agents will find and use relevant data and ignore data that is of no concern to them. These agents do not change the plan as adaptation agents do as they only have read power

to analyze how the plan works in the particular subject area. To have a subject area of expertise, these agents usually wrap or communicate with an outside knowledge source that specializes in that area. The weather agent for example has a weather feed it can communicate with and an understanding of weather rules and the weather capabilities of actors in the plan.

During the DEEP process, scoring critic agents use the adapted plans on the blackboard to evaluate. The agents will find the data they need in the plan to further their processing. For example the weather agent will find all the plan location data contained in the plan and then can use that location data to gain weather information using an external source, such as an RSS feed. Once the CPR plan object has been parsed for the needed data, the scoring agent will process. The implementation behind evaluation will be different for all scoring agents and so will the data required for them out of CPR. It is probable as new scoring agents are added CPR has to evolve to include more information. Once evaluation has finished, these agents have a scoring algorithm that will produce a score that is tied to that particular plan and is posted up to the blackboard.

These agents follow a general technical scheme. They implement the Java Agent Development Framework (JADE), register to and setup listeners to the blackboard, have a knowledge source either internally or externally, have an evaluation implementation, and a scoring algorithm. These agents can also use human sources as their knowledge base allowing for mixed initiative interaction.

The case-based reasoning system has produced plans and placed them on the blackboard where the initial adaptation agents updated the plans to the present context. Then the scoring critics evaluated the plans based on several domains of knowledge and posted scores, which now means the blackboard is full of plans that have been updated and scored. These plans now need to be evaluated and chosen to move on for final simulation. The deciding entity is an execution selection critic agent.

2.3.4.3 Execution Selection Critic Agents

Now that we have several plans which have been instantiated, adapted, and scored by the various agents in the DEEP system, a final agent is responsible for selecting the top scoring plan(s) and sending them off for execution. This agent is known as the Execution Selection Critic Agent, its specialization is taking all of the information on the blackboard and using it to evaluate and rank the plans. It then either decides which plan(s) are to be executed by either prompting the user for mixed-initiative input, or selecting it on its own.

The first challenge in developing the Execution Selection Agent was how to know when to notify the agent when the plans on the blackboard have converged. In other words, the problem was how an agent, who by design isn't dependent upon other agents, knows when other agents are done accomplishing their tasks. The solution chosen was to utilize the message passing which was discussed earlier. What happens is the Interface agents broadcast a message containing information about how many instantiated plans they are posting on the blackboard. When the Adaptation Agents realize they have work to do,

they notify the Execution Selection Agent to wait until their work is completed. Similarly, the Scoring Critic Agents notify the agent to wait as well. The Execution Selection Agent receives all these messages and uses them to determine when the appropriate number of adapted plans and scores are placed on the blackboard. The Execution Selection Agent then keeps a count of all the adapted plans and scores that are being placed on the blackboard and this is how it realizes when the plans have converged.

After the agent realizes that the plans have converged, it then sorts the plans in order of best to worst. In order to sort the plans, it uses the scores associated with each plan to rate them. This functionality does not currently assign a weight to the scores, however, in the future it will.

The Execution Selection Critic Agent also has an interface to pass its plans to simulators to simulate the outcome of certain plans. The idea behind these simulations is that these would be quick simulation engines that could run quickly, in parallel, and could help give insight about the plans to the selection agent. A more detailed simulation will take place when the top rated plans have been selected. The following section will discuss the more detailed simulation engines and how they will be used.

So the plans have been rated, sorted, and now they are ready to be executed. This is a great opportunity for mixed-initiative interaction with the system in that the specified number of plans could be displayed to the user for evaluation and validation. In our current proof-of-concept demonstration, the Execution Selection Agent sends a message to the Interface Agent containing the top rated plans. The user is then notified that they can view the top rated plans and select one to be sent off to the main simulation engine.

2.3.4.4 Simulation Critic Agents

After the Execution Selection agent has decided on the plans to be simulated, the DEEP architecture will run a much more powerful simulation against the plans to test and evaluate them. The CPR object that has been passed around throughout this cycle has contained plan information, but it also contains sections ready to contain the plan outcome and events. The plan outcome section is crucial to future reasoning over the plans due to the need to understand what happened while running that plan in the context it was set in. The simulator has to be able to produce an outcome with the results and how the results were obtained. This is important because it has to be able to document which objectives succeeded or failed and why. This information will populate the event section. The events section will capture the events that either allowed objectives to succeed or fail. An example might be the events leading up to the failure of a supply transport due to enemy interception. The events would describe the specific happenings of that situation including the enemy planes shooting down the tanker.

Simulation in the DEEP cycle presently involves a randomly generated outcome tied to specific plans, with a few different up and coming options to interchange in. The future work section will discuss the avenues that are open in the near future. DEEP simulation at the moment is setup as a placeholder simulation. The last section left off with the user selecting a plan to be sent off to the simulation engine. Once the simulation engine has

the plans to be simulated, simulations will occur and the sections of the plan, outcome and event, will be populated accordingly and it appears on the blackboard with updated data.

Once simulation is done, the operator can view the plans on the blackboard and make a user selection on whether or not any of the plans are viable. The cycle explained could occur again or could have been restarted earlier in the cycle. A plan can be replanned in DEEP if it is determined it is needed.

3 Future

As discussed earlier, this paper has documented the principle architecture designed for the DEEP project. The explanation of the architecture has also been supplemented by the current prototype proof-of-concept example implementation of several of the DEEP software components. The initial drive of the DEEP project was to build a research platform that was developed in a modular fashion allowing these future advancements to be retrofitted in very easily. Discussed below are some of the definitive goals that have been established for DEEP as future research areas where the DEEP system shall improve.

3.1 Distributed Database

One of the goals in the near future is to integrate DEEP with other systems. One viable approach to integrating with other systems is interfacing the blackboard with their data repository and utilizing it as the blackboard data store.

One integration effort has been to replace the data store component of the blackboard with an Oracle database and leverage its distributed functionality. By utilizing this commercial database system, we can offload a lot of the issues inherent with a distributed system such as transaction support. Work has already begun with this and has been successful thus far. The major challenge with this is mapping the very deep, recursive object-oriented CPR data structure. To help address this challenge we decided to use the Object-Relational mapping tool Hibernate. This tool allows the blackboard to more fluidly and easily post and retrieve objects from the database. With a connection to an Oracle database management system, not only will we be able to utilize it as the blackboard data store, it will also provide DEEP with a popular integration medium for future integration efforts.

3.2 Formalized Messaging Structure

The messaging system currently employed is a simple messaging scheme containing a few different message types that can be sent that are based on a taxonomy. One of the reasons to improve the messaging system is to separate the messaging structure away from current components so it may be updated and improved as its own component, having minimal impact on the system. The future vision is to leverage research done in speech act theory and formalize a message structure for communication between the various components of the DEEP system. The JADE Agent Communication Language

(ACL) does have some grounds in speech act, but the idea is to go beyond relying on the structure of JADE and have a structure anything in DEEP can adhere to. The idea is the DEEP team will develop a semantically driven language for the components to communicate using. The key is that the communicating entities will in theory be involved in a communication of sorts where negotiations can happen. By designing a separate and formalized messaging structure in the DEEP project, it will lead to a more robust architecture to be used as a research platform.

3.3 Multi-Case Distributed Planning

A major challenge to be researched is multi-case distributed planning. The key challenge here is how multiple cases are fused into one coherent plan.

3.4 Semantic CPR

CPR is being updated by members of the DEEP team looking to semantically enrich the plan representation. Because of the distributed aspect of DEEP, which involves humans and machines, the need for a proper understanding and consensus on terms is very important. Given the amount of planning systems and differing experiences that the DEEP system will have available to it, the ability to capture and express these different experiences and their meanings concisely is required. Semantically enriching the CPR is necessary in working towards the goal of expressing and understanding intended meaning. Presented in this same conference is a paper discussing this topic exclusively and goes into much greater detail. For more information, please refer to "Semantic Interoperability in Distributed Planning" by Staskevich, Hudack, Lawon, and Carozzoni.

3.5 Simulation Technologies

Earlier in the report, the simulation agent was discussed and it was explained how the simulation environment used in DEEP is a placeholder for future simulation technologies. For proof-of-concept, the current simulation produces a randomly generated outcome for the plan being simulated. There are multiple options for upgrading this area and the DEEP team is already researching different simulation technologies. One is a war gaming simulator called Modern Air Power. Another is a contract funded simulation engine evaluating game theory. Other possibilities include in-house modeling and simulation activities that could present themselves to be of good use. Modern Air Power has already been evaluated and worked with, but has yet to become fully integrated into the DEEP architecture due to timing constraints. The other two simulation engines have yet to be evaluated or integrated by the DEEP team.

Modern Air Power was the initial simulation engine being added to the DEEP platform. This engine was more tuned to tactical simulation; though it has the ability to model the effect based operations of a given plan. An effort to develop an adversarial based agent simulation engine is underway and time will tell if it is the simulation engine of choice as it is not fully complete yet.

One or more of these simulation technologies will end up as the DEEP simulation engine to give the DEEP system the ability to evaluate plans and populate the simulated plans

with outcomes and events. This information will then be re-assimilated into the casebase, which will allow for a self population of new cases into the system once underway.

4 Conclusion

In October 2007, the DEEP project completed year one of its scheduled four-year schedule. The objective of the first year was successfully completed – development of a "research platform" to further support more aggressive research in the areas of distributed C2 and analogical reasoning, and how to apply the technology to advance the state of C2.

The experience-base of the current DEEP prototype was based on the Battle for Guadalcanal (Aug 42 –Feb 43) and focused on joint warfighting versus being air war oriented. There was strong rationale for using Guadalcanal as the experience-base:

- It was an intense and full spectrum combined land/air/sea campaign
- It had the first JFACC-like position (Operation Watchtower)
- It had both traditional and asymmetric aspects
- It included issues addressing cultures and technologies
- It leveraged coalitions and alliances (i.e. Australians)
- It was thoroughly documented and analyzed, a key aspect to facilitate case validation and verification

For the scenario, a totally different venue was chosen, Operation Unified Assistance, the U.S. response to the earthquake just north of Sumatra Island and the resulting tsunami. Currently, DEEP demonstrates the power of analogical reasoning in using the historical, military-focused Guadalcanal experience-base to plan for a modern humanitarian relief operation. DEEP will be used to experiment with the capability of analogical reasoning to improved planning speed, plan quality, and plan creativity, with the vision of becoming a truly distributed planning capability in the future.

5 References

A.Helsinger, M. Thome, T. Wright. "Cougaar: A Scalable, Distributed Multi-Agent Architecture." In proceedings of IEEE SMC04 at The Hague.

AF/A5 Plans. "AF C2 Enabling Concepts". May 2006

Alberts, D. and R. Hayes. "Planning: Complex Endeavors", 2007 page 217,

Bellifemine, Fabio. "JADE ADMINISTRATOR'S GUIDE." November 10, 2006. JADE 3.4.1

Bellifemine, Fabio. "JADE PROGRAMMER'S GUIDE." August 21, 2006. JADE 3.4

Caire, Giovanni. "JADE TUTORIAL JADE PROGRAMMING FOR BEGINNERS." December 4, 2003. JADE 3.1

Corkill, Daniel D., Blackboard architectures and control applications. In: Proceedings 5 IEEE International Symposium on Intelligent Control 1990, IEEE, Piscataway, NJ (1990), pp. 36–38

Corkill, Daniel D., Blackboard Systems. AI Expert, 6(9):40-47, September, 1991.

Corkill, Daniel D., Collaborating software: Blackboard and multi-agent systems & the future. In Proceedings of the International Lisp Conference, New York, New York, October 2003.

Hammond, Kristian J. Case-Based Planning: A Framework for Planning from Experience. Cognitive Science 14, 1990. pp. 385-443.

Pease, R. Adam. "Core Plan Representation." Version 4 November 6, 1998

Twitchell, Douglas P. "Using Speech Act Theory to Model Conversations for Automated Classification and Retrieval." June 2-3, 2004.